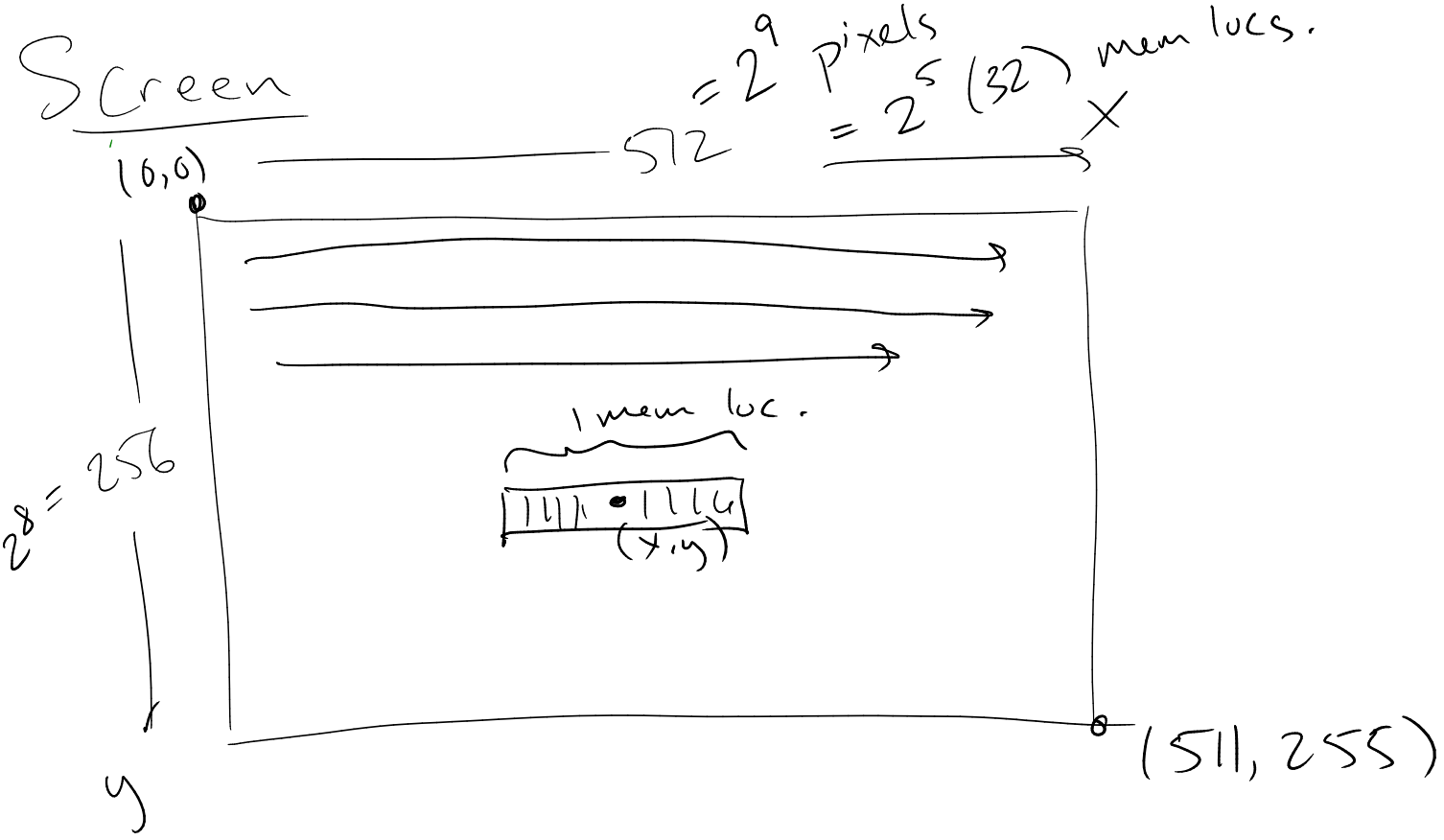
Screen



drawPixel(x, y).

pixel (x,y) is pixel # $2^9 y + x$.

($2^9 y$: complete rows above (x,y); x: row up to (x,y).)

Hence it is memory loc

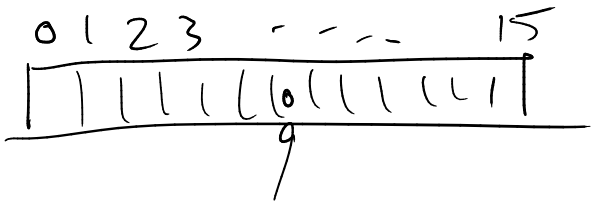
$$(2^9 y + x)/16 = 2^5 y + x/16.$$

ie. we want memory address

SCREEN + 32y + x/16

= SCREEN[32y + x/16].

Within its 16-bit memory location,
(x,y) is pixel # x%16.



(x,y)

Note: x%16 is same as x & $000\cdots01111_2$

= x & 15.

- Use Memory.peek to read contents of memory loc.
- Set bit (x%16).
- Write it back w/ Memory.poke.

How to set bit i in a 16-bit value?

- To set bit i to 1: $v = v \mid 2^i$

v: 1011000110...1

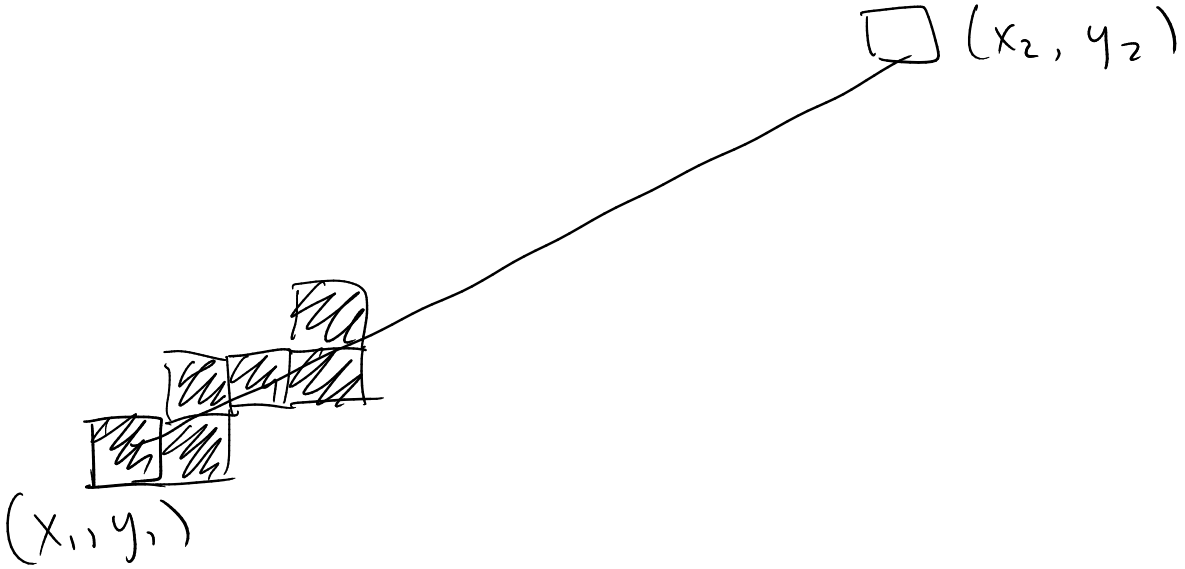
2^i: 0000010000...0

v

1

- To set bit i to 0: $v = v \& (N\ 2^i)$

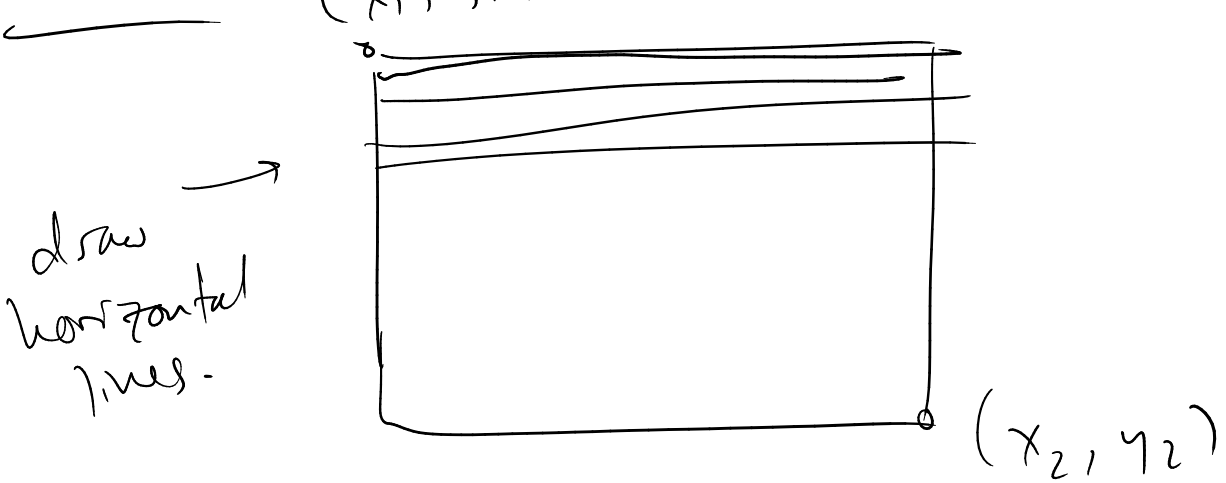
drawLine



Bresenham's algorithm.

- Careful to handle all directions
- Special cases for horiz and/or vertical?

drawRect-

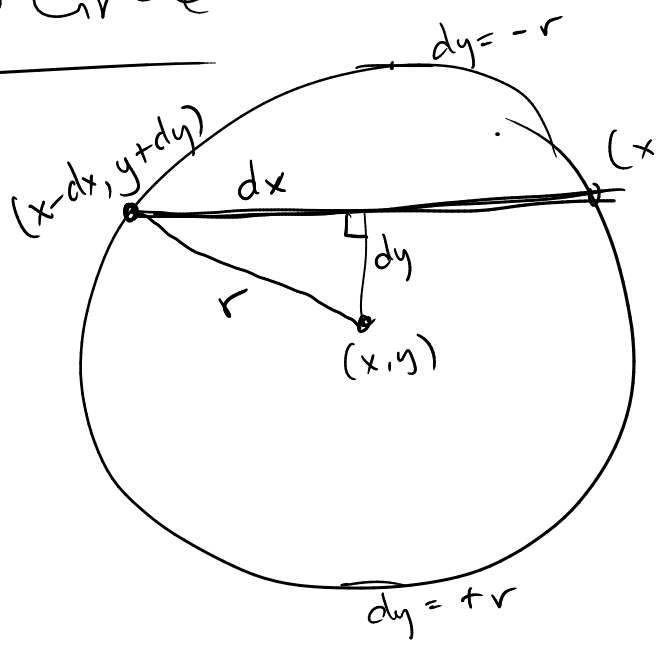


Level 1: call drawLine.

Level 2: special helper to draw horiz. line, loop through x coords & call drawPixel.

Level 3: optimized horiz. line that sets all 16 pixels in each mem loc at once when possible.

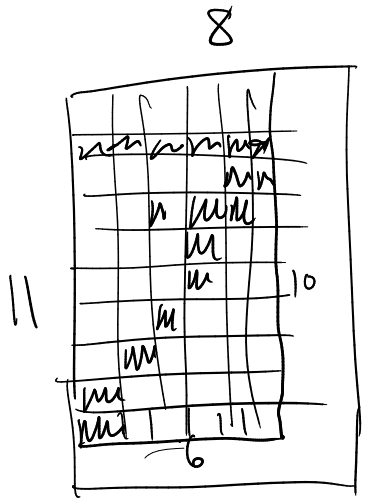
drawCircle



$dx^2 + dy^2 = r^2$

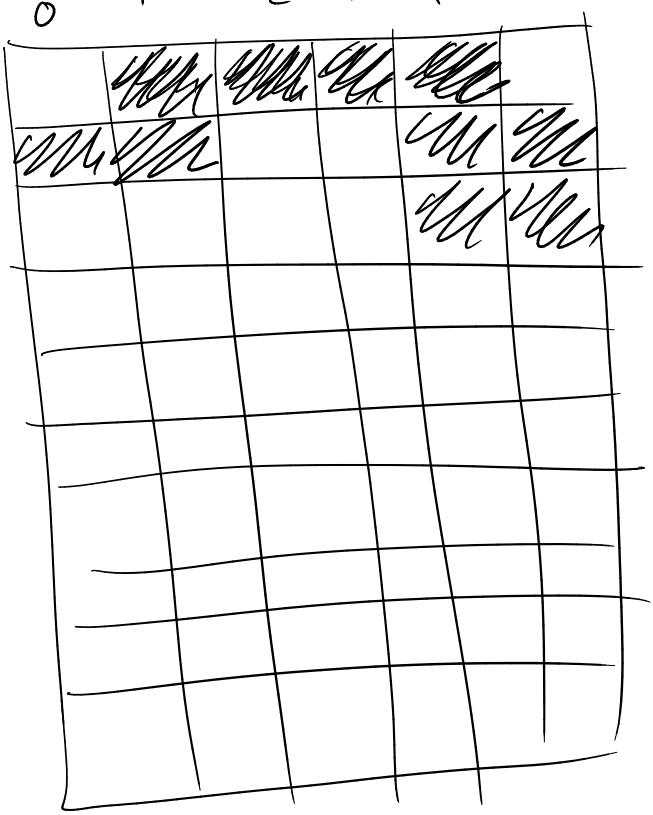
$\Rightarrow dx = \sqrt{r^2 - dy^2}$

Output -



30 = 16 + 8 + 4 + 2

00011110



51 =

32 + 16 + 2 + 1

00110011

48 =

32 + 16

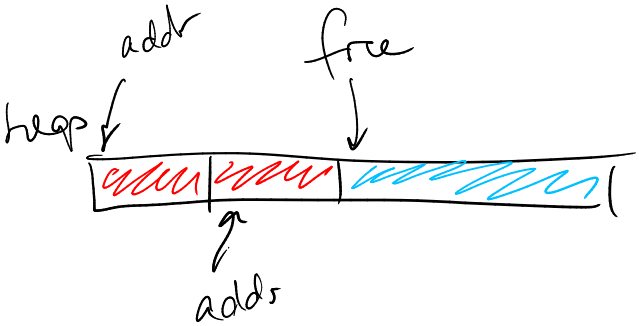
00110000

- Make a helper function to draw character @ current location.

  2 nested loops, call drawPixel

  or set all 8 bits at once for each row.

Memory



① Naive version.

free = 2048.

```
alloc(size):
    free += size
    return old value of free.
deAlloc(addr):
    do nothing.
```